**IN THE CLAIMS**

For the convenience of the Examiner, all pending claims of the present Application are shown below whether or not an amendment has been made.

Please amend the claims as follows.

1. **(Currently amended)** A method of detecting viral code in subject files, comprising:

creating an artificial memory region spanning one or more components of the operating system;

emulating execution of at least a portion of computer executable code in a subject file;

**monitoring operating system calls by the emulated computer executable code to detect** ~~**detecting**~~ an attempt by the emulated computer executable code to access the artificial memory region; and

determining based on the attempt to access the artificial memory region that the emulated computer executable code is viral.

~~**; and**~~

2. **(Canceled)**

3. **(Canceled)**

4. **(Previously presented)** The method of claim 1, further comprising:

emulating functionality of an identified operating system call while monitoring the operating system call to determine whether the computer executable code is viral.

5. **(Canceled)**

6. **(Canceled)**

7. **(Canceled)**

8. **(Original)** The method of claim 1, further comprising monitoring access by the emulated computer executable code to dynamically linked functions.

9. **(Previously presented)** The method of claim 8, wherein the artificial memory region spans a jump table containing pointers to the dynamically linked functions.

10. **(Currently amended)** A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps for detecting viral code in subject files, the method steps comprising:

creating an artificial memory region spanning one or more components of the operating system;

emulating execution of at least a portion of computer executable code in a subject file;

**monitoring operating system calls by the emulated computer executable code to detect** ~~**detecting**~~ an attempt by the emulated computer executable code to access the artificial memory region; and

determining based on the attempt to access the artificial memory region that the emulated computer executable code is viral.

11. **(Currently amended)** A computer system, comprising:

a processor; and

a program storage device readable by the computer systems, tangibly embodying a program of instructions executable by the processor to perform method steps for detecting viral code in subject files, the method comprising:

creating an artificial memory region spanning one or more components of the operating system;

emulating execution of at least a portion of computer executable code in a subject file;

**monitoring operating system calls by the emulated computer executable code to detect** ~~**detecting**~~ an attempt by the emulated computer executable code to access the artificial memory region; and

determining based on the attempt to access the artificial memory region that the emulated computer executable code is viral.

12. **(Currently amended)** A computer data signal embodied in a transmission medium which embodies instructions executable by a computer for detecting in a subject file viral code that uses calls to an operating system, the signal comprising:

a first segment comprising CPU emulator code, wherein the CPU emulator code emulates execution of at least a portion of computer executable code in the subject file;

a second segment comprising memory manager code, wherein the memory manager code creates an artificial memory region spanning components of the operating system; and

a third segment comprising monitor code, wherein the monitor code **monitors operating system calls by the emulated computer executable code to dete ct** ~~**det ects**~~ attempts by the emulated computer executable code to access the artificial memory region and determines based on an attempt to access the artificial memory region that the emulated computer executable code is viral.

13. **(Previously presented)** The computer data signal of claim 12, further comprising:

a fourth segment comprising analyzer code, wherein the analyzer code emulates functionality of the identified operating system call to determine whether the computer executable code is viral.

14. **(Currently amended)** An apparatus for detecting in a subject file viral code that uses calls to an operating system, comprising:

a CPU emulator;

a memory manager component that creates an artificial memory region spanning one or more components of the operating system and that creates a custom version of an export table, wherein the custom version of the export table is associated with a plurality of entry points and wherein the entry points comprise predetermined values; and

a monitor component, wherein the CPU emulator emulates execution of at least a portion of computer executable code in the subject file, and the monitor component:

**monitors operating system calls by the emulated computer executable code to detect** ~~**detecting**~~ an attempt by the emulated computer executable code to access the artificial memory region; and

determines based on the attempt to access the artificial memory region that the emulated computer executable code is viral.

15. **(Previously presented)** The apparatus of claim 14, further comprising:

an auxiliary component; and

an analyzer component,

wherein the auxiliary component emulates functionalities of an identified operating system call, and the monitor component monitors the operating system call to determine whether the computer executable code is viral, while emulation continues.

16. **(Previously presented)** The apparatus of claim 15, wherein the auxiliary component emulates functionalities of the operating system call.

17. **(Canceled)**

18. **(Canceled)**

19. **(Canceled)**

20. **(Original)** The apparatus of claim 14, wherein the artificial memory region created by the memory manager component spans a jump table containing pointers to dynamically linked functions, and the monitor component monitors access by the emulated computer executable code to the dynamically linked functions.

21. **(New)** The method of claim 1, further comprising:

monitoring accesses by the emulated computer executable code to the artificial memory region to detect looping; and

determining based on a detection of looping that the emulated computer executable code is viral.

22. **(New)** The method of claim 1, wherein creating an artificial memory region comprises creating a custom version of an export table with predetermined values for the entry points.

23. **(New)** The method of claim 1, further comprising:

monitoring access by the emulated computer executable code to dynamically linked functions; and

determining based on attempted access to dynamically lined functions that the emulated computer executable code is viral.